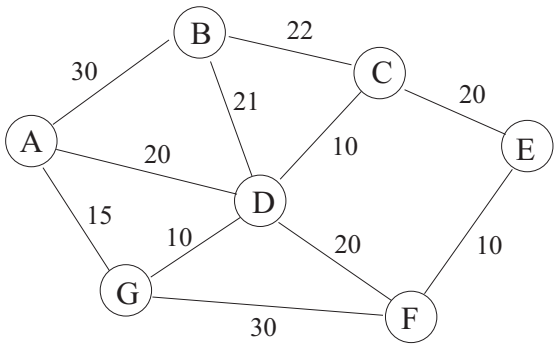
Solved exercises for the course of

Foundations of Operations Research

Roberto Cordone

Optimal spanning tree problem

A provider of fiberoptic connectivity wants to establish a fiberoptic network among a set of seven places, indicated by the letters from A to G. For technical reasons, the links must follow the road network described in the following figure. The value on each road provides its length in km.



The network must be built with the minimum total expense. The cost of building a link is $K = 100\,000$ euros/km, except for the links which connect place E to the adjacent ones, for which the cost is 20% larger. Moreover, F and G are already connected by a link owned by the provider, whereas C and D are connected by a competitor, who is keen to sell the link for a total cost equal to $C = 1\,200\,000$ euros.

Formulate the problem as a graph optimization model.

Solve the model with an appropriate algorithm.

Solution

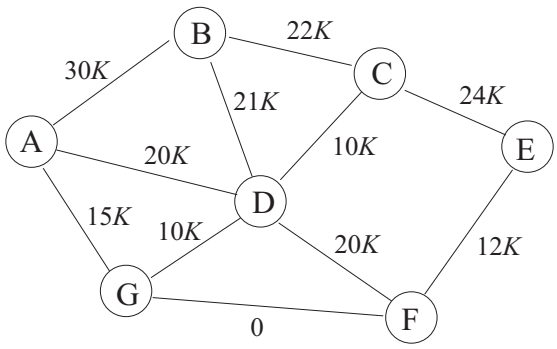
The figure represents as a graph the map of the land on which the network must be established. This makes it obvious that we are facing a graph optimization problem, and that the solution consists of a suitable subgraph, having minimum cost and respecting the constraints imposed by the problem.

Since we are linking a set of places represented as vertices, choosing the links among a set of potential connections represented as edges, we are looking for a connected subgraph. Since all places must be connected, the subgraph is spanning. The links are bidirectional and symmetrical, so that the graph is undirected, just as the road network. Finally, it is quite clear that the required subgraph must be acyclic, because otherwise the cycles could be broken removing edges: the solution would still be connected, and less expensive. Therefore, cyclic solutions are feasible, but certainly not optimal. We can consider them as unfeasible.

Since we are looking for a connected acyclic spanning subgraph, the correct model of the problem is the *minimum spanning tree problem.*

Besides a graph topology, this problem requires to associate a cost function to the edges. The text of the problem does not provide the costs required to build the fiberoptic links, but it provides the length d_e for each potential link and the cost per km, K. Most of the time, the cost required to activate a link is equal to Kd_e. However, on the edges of $\Delta_{\{E\}}$, i. e. the edges incident to vertex E, the cost must be increased by 20%. Moreover, link (F, G) is given for free, whereas link (C, D) can be built from scratch, with the cost defined above, or bought from a competitor, with a cost equal to C. Of course, one will choose the less expensive between the two choices, which is the former. The cost of each potential link, therefore, is:

- $c_{ij} = Kd_{ij}$ for each edge $(i, j) \in E \setminus \{(C, E), (F, E), (G, F)\}$
- $c_{ij} = 1.2Kd_{ij}$ for the edges $(i, j) \in \{(C, E), (F, E)\}$
- $c_{ij} = 0$ for edge $(i, j) = (G, F)$
- $c_{ij} = \min(Kd_{ij}, 1\,200\,200)$ for edge $(i, j) = (C, D)$



We are not required to provide the mathematical programming formulation. Should we, we could easily guess that the decision variables of the problem are binary variables indicating for each edge whether it belongs or not to the solution. The problem can be solved applying Prim's algorithm or Kruskal's algorithm on graph $G = (V, E)$ with cost function c_e.

Prim's algorithm

The intuitive idea of this algorithm is to start with the simplest possible acyclic connected subgraph: a single vertex v_{in} chosen arbitrarily, and to approach a spanning subgraph, modifying the partial solution step by step at minimum cost.

$Prim(V, E, c, v_{\text{in}}, X)$

$S := \{v_{\text{in}}\}$; $X := \emptyset$;

While $S \subset V$ **do**

$[u, v] := \arg \min_{u \in S, v \notin S} c_{uv}$;

$X := X \cup \{[u, v]\}$;

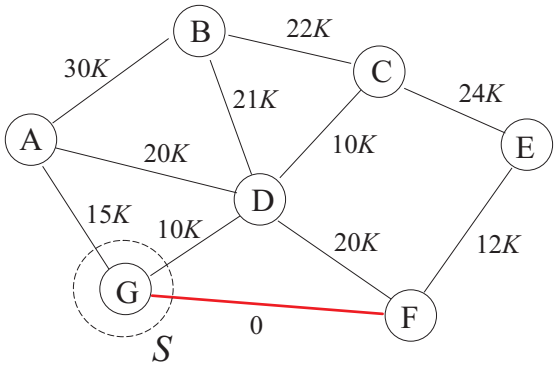
$S := S \cup \{v\}$;

EndWhile

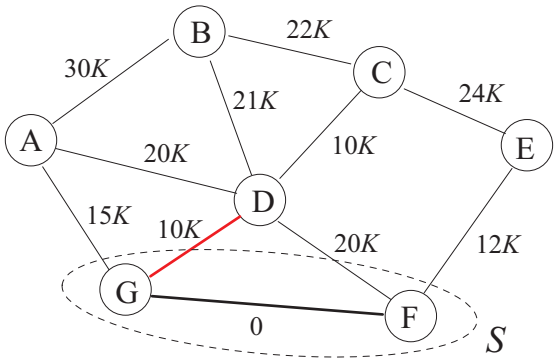
Return $T\,(V, X)$

At each step, one selects the minimum cost edge in the cut induced by the subset S of the vertices already connected. Then, one adds the edge to the tree, together with the extreme vertex previously not connected to S. Only the edges in the cut are considered because the edges of E_S (that is, between vertices internal to S) would produce cycles, while the edges between vertices external to S would produce a disconnected subgraph.

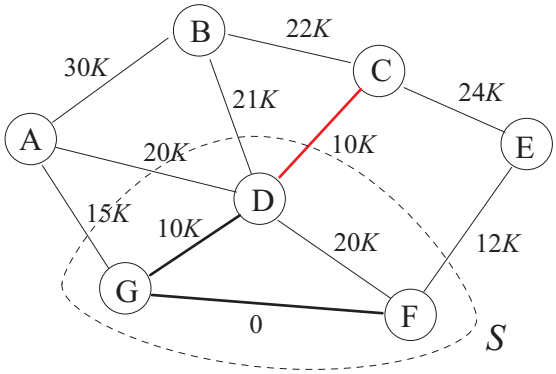
Iteration 1 Let us assume that the starting vertex be $v_{\text{in}} = G$. The choice is between edges (F, G) ($c_{FG} = 0$), (D, G) ($c_{DG} = 10K$) and (A, D) ($c_{FG} = 15K$). The cheapest is the first one, which is accepted: $(F, G) := \arg\min_{u \in S, v \notin S} c_{uv}$, so that $X := \emptyset \cup \{(F, G)\}$ and $S := \{G\} \cup \{F\}$.



Iteration 2 The cut induced by S contains edges (A, G) (with $c_{AG} = 15K$), (D, G) (with $c_{DG} = 10K$), (D, F) (with $c_{DF} = 20K$) and (E, F) (with $c_{EF} = 12K$). The cheapest one is (D, G), so that $X := \{(F, G), (D, G)\}$ e $S := \{D, F, G\}$.

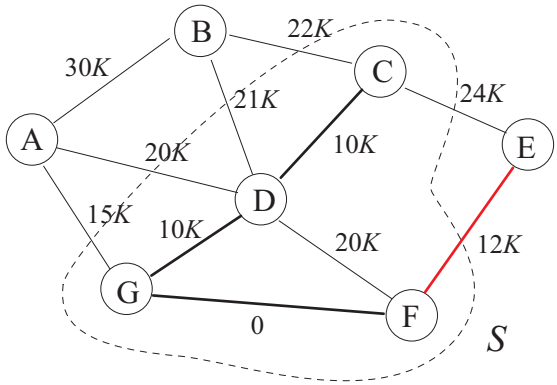


Iteration 3 The cut induced by S contains edges (A, G) (with $c_{AG} = 15K$), (D, G) (with $c_{DG} = 10K$), (D, F) (with $c_{DF} = 20K$) and (E, F) (with $c_{EF} = 12K$). the cheapest one is (D, G), so that $X := \{(F, G), (D, G)\}$ e $S := \{D, F, G\}$.

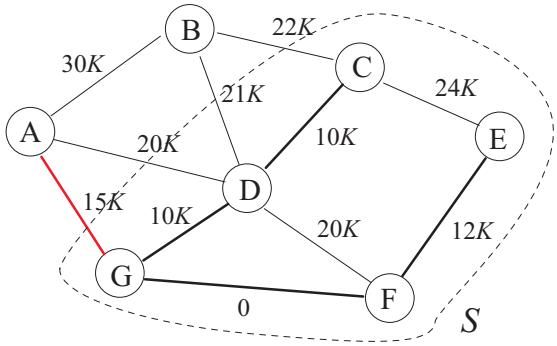


Iteration 4 The cut induced by S contains edges (A, G) (with $c_{AG} = 15K$), (D, G) (with $c_{DG} = 10K$), (D, F) (with $c_{DF} = 20K$) and (E, F) (with $c_{EF} =$

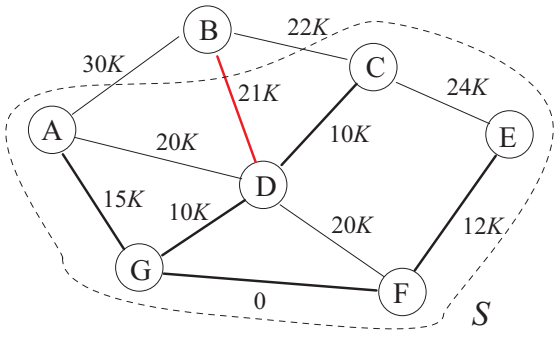
$12K$). The cheapest one is (D, G), so that $X := \{(F, G), (D, G)\}$ and $S := \{D, F, G\}$.



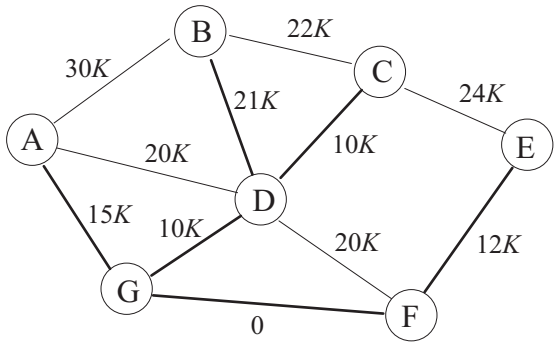
Iteration 5 The cut induced by S contains edges (A, G) (with $c_{AG} = 15K$), (D, G) (with $c_{DG} = 10K$), (D, F) (with $c_{DF} = 20K$) and (E, F) (with $c_{EF} = 12K$). The cheapest one is (D, G), so that $X := \{(F, G), (D, G)\}$ e $S := \{D, F, G\}$.



Iteration 6 The cut induced by S contains edges (A, G) (with $c_{AG} = 15K$), (D, G) (with $c_{DG} = 10K$), (D, F) (with $c_{DF} = 20K$) and (E, F) (with $c_{EF} = 12K$). The cheapest one is (D, G), so that $X := \{(F, G), (D, G)\}$ e $S := \{D, F, G\}$.



Termination Now, the subgraph is spanning ($S = V$), so that the algorithm terminates. The following figure reports he result, whose total cost is $c_X = \sum_{ij} c_{ij} x_{ij} = 6\,800\,000$ euros.



Kruskal's algorithm

The intuitive idea of this algorithm is to start with the simplest possible acyclic spanning subgraph: the whole vertex set V, with no edge at all, and to approach a connected subgraph modifying the partial solution step by step at minimum cost.

Kruskal(V, E, c, X)

Sort E by nondecreasing costs: e_{π_1}, e_{π_1}, ..., e_{π_m}

$X := \emptyset$;

For $i := 1$ **to** m **do**

    If $(X \cup e_{\pi_i})$ does not contain cycles **then** $X := X \cup \{e_{\pi_i}\}$;

EndFor

Return $T\,(V, X)$

Once the edges are sorted by nondecreasing costs, each step of the algorithm considers the current edge and adds it to the solution, provided that this does not induce any cycle with the edges added previously.

An efficient implementation

In order to verify acyclicity, it is useful to maintain an efficient representation of the connected components of the graph. Such a representation could consist of a simple vector, associating each vertex of the graph to the index of the component which includes it, or to the index of a "representative" vertex of that component.

This data structure would allow to determine in constant time whether a new edge induces cycles. In that case, in fact, the two extreme vertices of the new edge would be already connected by a path, and therefore would belong to the same connected component. The drawback of such a representation is that, after adding a new edge, the two corresponding components should be merged into a single one, and the index of all the vertices in at least one of them should be updated. Such an update would require $O(n)$ time. Consequently, the time gained in the acyclicity test would be completely lost in the update of the data structure.

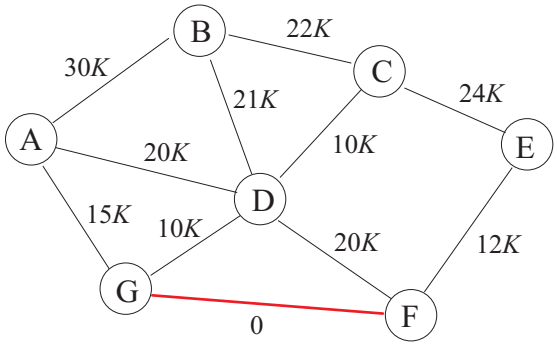
A more efficient solution is, on the contrary, to define a vector associating each vertex of the graph not directly to the representative vertex, but to another vertex of the graph, which could refer to a third one, and so on, until the representative vertex is reached. This yields a tree structure, which allows to merge two components in a trivial way, when their representative vertices are known: one of them must be simply "appended" to the other writing the index of the other in the vector. This can be done in constant time. For example, in order to merge the component represented by vertex B with the component represented by vertex F, one simply writes F in position B of the vector. The acyclicity test becomes slightly different, because it requires to follow two chains of references, starting from the extreme vertices of the new edge to the representative vertices of the two components. It can be proved that, if we also save in another vector the number of elements of each component, and if we always append to smaller component to the larger one, the reference chains keep limited with respect to the size of the graph. To save space, one can even save the number of elements in the same vector as the references, in the position corresponding to the representative vertices (with a negative sign, to distinguish them).

Initialization The edges of the graph, sorted by nondecreasing costs, are: (F, G), (C, D), (D, F), (E, F), (A, G), (A, D), (D, F), (B, D), (B, C), (C, E), (A, B). Notice that some edges have the same cost, and their order is completely arbitrary.

The auxiliary vertex is initialized so as to describe the fact that each vertex is a connected component of cardinality 1, represented by the vertex itself. Therefore, the vector contains -1 in each position, to signify that the vertex is a representative (negative sign) and that the cardinality of the component is 1.

A	B	C	D	E	F	G
-1	-1	-1	-1	-1	-1	-1

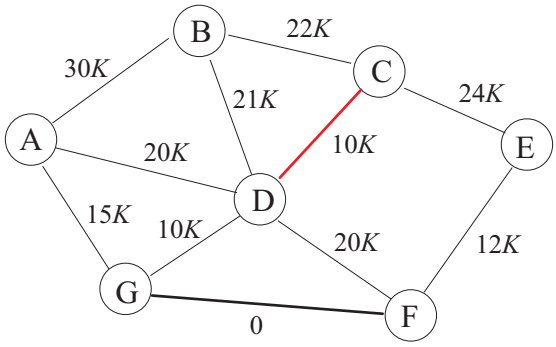
Iteration 1 The cheapest edge is (F, G), with cost $c_{FG} = 0$. It does not induce cycles with $X = \emptyset$, and therefore is directly added to the solution. In fact, the auxiliary vector contains negative values in positions F and G, suggesting that the two vertices are representatives of different connected components.



After adding the new edge, the auxiliary vector is updated appending component $\{G\}$ to component $\{F\}$. The two components have the same cardinality, so that the choice is arbitrary. In order to perform the append, one writes F in position G and writes the (negative) sum of the two cardinalities $-1-1=-2$ in position F:

A	B	C	D	E	F	G
-1	-1	-1	-1	-1	-2	F

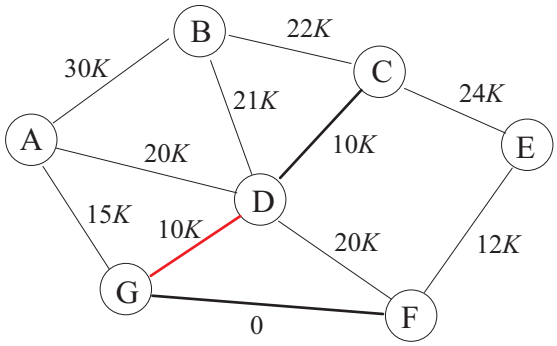
Iteration 2 The second edge is (C, D), with cost $c_{CD} = 10K$. It does not induce cycles, because the positions corresponding to C and D both contain negative values, so that the two vertices belong to different connected components. Therefore, the edge is added to the solution.



The auxiliary vector is updated appending component $\{D\}$ to component $\{C\}$, once again with an arbitrary choice, writing C in position D and the (negative) sum of the two cardinalities $-1-1=-2$ in position C.

A	B	C	D	E	F	G
-1	-1	-2	C	-1	-2	F

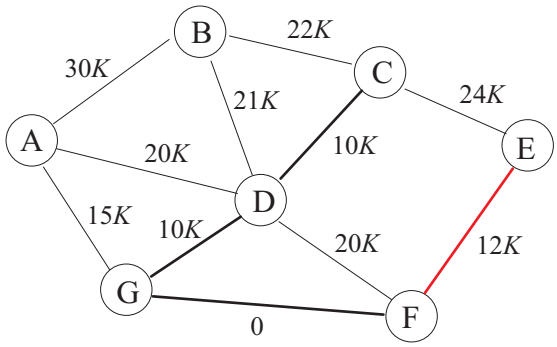
Iteration 3 The third edge is (D, G), with cost $c_{DG} = 10K$. It does not induce cycles, because vertex D refers to vertex C, which is a representative (the vector contains a negative value in position C). Vertex G refers to vertex F, which is also a representative. Since the two representatives are different, the two vertices belong to different connected compnonents, the edge induces no cycles and it is added to the solution.



The auxiliary vector is updated appending component $\{F, G\}$ to component $\{C, D\}$, once again with an arbitrary choice (same cardinality), and writing C in position F and the (negative) sum of the two cardinalities $-2 - 2 = -4$ in position C.

A	B	C	D	E	F	G
-1	-1	-4	C	-1	C	F

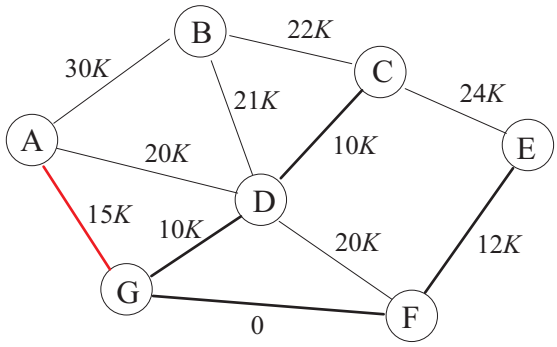
Iteration 4 The fourth edge is (E, F), with cost $c_{EF} = 12K$. It does not induce cycles, so that it is added to the solution. In fact, vertex E is an isolated component, while vertex F refers to vertex C, which is a representative.



The auxiliary vector is updated appending component $\{E\}$ to component $\{C, D, F, G\}$, since the former has smaller cardinality (one vertex versus four). We write C in position E and the (negative) sum of the two cardinalities $-4-1 = -5$ in position C.

A	B	C	D	E	F	G
-1	-1	-5	C	C	C	F

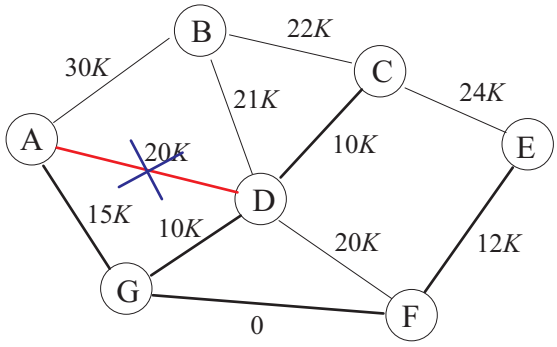
Iteration 5 The fifth edge is (A, G), with cost $c_{AG} = 15K$. It does not induce cycles, so that it is added to the solution. In fact, vertex A is isolated, whereas vertex G refers to vertex F, which refers to the representative vertex C.



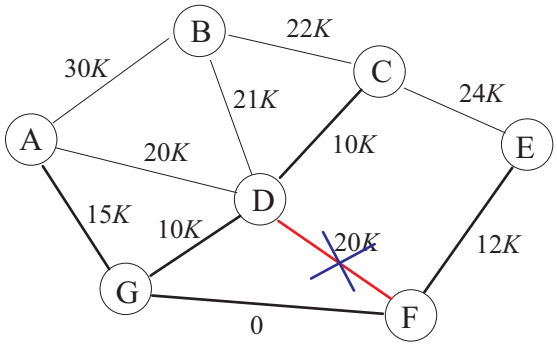
The auxiliary vector is updated appending component $\{A\}$ to component $\{C, D, E, F, G\}$, since the former is smaller (one vertex versus five). We write C in position A and the (negative) sum of the two cardinalities $-5 - 1 = -6$ in position C.

A	B	C	D	E	F	G
C	-1	-6	C	C	C	F

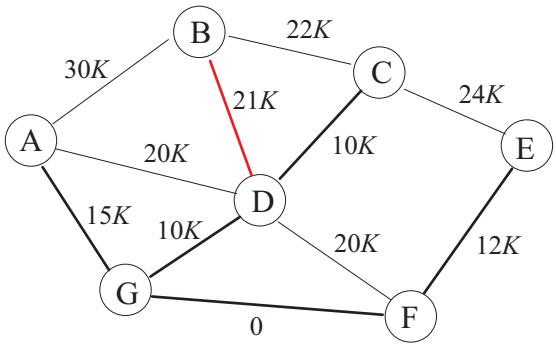
Iteration 6 The sixth edge is (A, D), with cost $c_{AD} = 20K$. It induces cycle (A, D, G), so that it is discarded. In fact, vertices A and D both refer to representative C.



Iteration 7 The seventh edge is (D, F), with cost $c_{DF} = 20K$. It induces cycle (D, F, G), so that it is discarded. In fact, vertices D and F both refer to representative C.



Iteration 8 The eighth edge is (B, D), with cost $c_{BD} = 21K$. It does not induce any cycle, so that it is added to the solution. In fact, vertex B is isolated, whereas vertex D refers to vertex C, which is a representative.

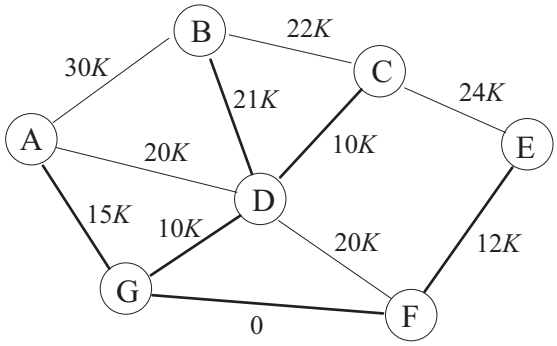


The auxiliary vector is updated appending component $\{B\}$ to component $\{A, C, D, E, F, G\}$, since it is smaller (one vertex versus six). We write C in position B and the (negative) sum of the two cardinalities $-6 - 1 = -7$ in position C.

A	B	C	D	E	F	G
C	C	-7	C	C	C	F

Si noti come quasi tutti i vertici puntino direttamente al capocomponente C, riducendo fortemente il tempo necessario a determinare se due di loro sono connessi o no.

Termination We could proceed with the following edges, finding that all of them induce cycles. However, the current solution is already spanning and connected, since it contains $n = 7$ vertices and $mn - 1 = 6$ edges and it is acyclic by construction. Therefore, the algorithm terminates. The following figure reports the solution, whose total cost is $C = \sum_{ij} c_{ij} x_{ij} = 6\,800\,000$ euros.


B
22K
30K
C
21K
24K
A
20K
10K
E
D
15K
10K
20K
12K
G
F
0